Secure static content delivery for content distribution network using blockchain technology

Pier Paolo Tricomi
Student ID: 1179740

***Abstract*—A Content Distribution Network (CDN) is a new kind of network with the goal to distribute services and contents spatially relative to end-users to provide high availability and high performance. Several replicas of the Origin Server are used to reach this goal, but trust issues are now involved both between servers and among client and server. In this work is presented a new method to provide secure static content delivery using blockchain, a growing technology with the capability to ensure reliability and trust without a central authority.**

Moreover, a prototype of the sistem has been developed on Ethereum private network, in order to test the feasibility. The test shows the goodness of the system, and the possibility to create a new content distribution model on the Internet.

1. Introduction

Contents distribution is one of the most important aspects of the Internet. To improve the distribution the Content Distribution Networks (CDNs) are born. CDNs provide high availability and high performance because many replicas (Edge Servers) of the Origin Server are spatially distributed to faster fulfill a request. Many services like CoralCDN [1] CoDeeN [2] provide the possibility to create a CDN starting from the Origin Server, replicating contents and distributing them to end users from the best Edge Server, likely the geographically nearest. A typical scenario can be seen in Figure 1, the Origin Server has more replicas to distribute contents.

In this scenario, two new trust issues are involved. First of all, an attacker could modify the contents from the Origin Server to the Edge Server, thus if the indirectly attacked Edge Server serves that modified content it will be labeled as a misbehaving replica. Secondly, if an Edge Server is not directly managed by the owner of the Origin Server, it can serve different content like stale content or outright modified content, adding ads for example. Furthermore in this architecture the Origin Server is a single point of failure. If the Origin server is compromised all the Servers will misbehave.

The main contribute of this work is two-fold.

First, it is suggested a new architecture to overcome the trust issue between servers, mainteining the CDN properties. Second, the proposed system provides a secure method to deliver static contents, ensuring the integrity with small effort on clients and Network.

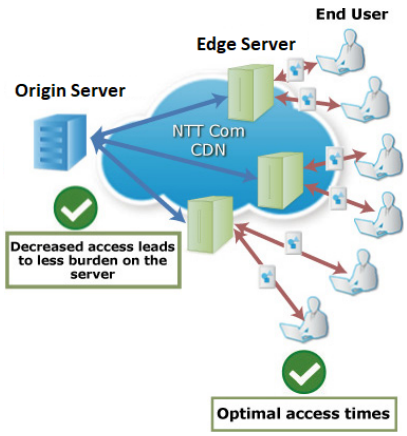

Figure 1. Typical CDN scenario where Origin Server has many Edge Servers to fulfill the clients requests.

2. Related Works

Checking the integrity of data retrieved from untrusted servers is a crucial problem nowadays. The typical approach for preventing contents tampering between clients and servers is to encrypt the end-to-end connection, for example, using the SSL protocol. This negates the functionality of the CDN, because the connection always requires the Origin Server, and the network is no more distributed. For static content, Merkle tree authentication [3] is a possible solution, where the server signs the content which is verified by the client. Also digital rights management schemes allows clients to verify data from untrusted server [4]. For dynamic content, [5] [6] propose the use of XML-based rules for managing the content, but it has to be limited to be easily verified by a client.

In peer-to-peer CDNs the problem is even more significant. LOCKSS [7] uses voting system for content integrity. Repeate the execution to detect misbehavior has been used in Rx [8] and in Vigilante [9] to discover bugs and worms respectively. These approachs require significant overhead on the client. In Pioneer [10] the verify effort is on the dispatcher, a trusted platform, but since the proof of correctness is extremely time sensitive it is not suitable for large scale systems. Finally Repeate and Compare system [11] is for both static and dynamic contents, it requires the repetition of the content to another replica and compares the results

to detect misbehaving replicas. However, the process has significant overhead, and the verify requests may overwhelm the network, thus only a fraction of the contents are verified. In this work blockchain technology is used. The structure of blockchain ensure trust between untrusted nodes without central authority. A system of contents distribution built over blockchain inherits all its benefits, making simple to share contents and verify them between clients and servers.

3. Blockchain Technology

After the rise of Bitcoin [12], its underlying structure, the blockchain technology, has been applied to a variety of usecases ranging from authentication [13] (using Ethereum and Smart Contracts) to medical reports [14]. Blockchain is based on append-only ledger, a growing list of records (called blocks) which are linked and secured using cryptography. The ledger can be viewed by all participating nodes and the updates are permitted only after the consensum of the network. Each block typically contains a cryptographic hash of the previous block, a timestamp and transaction data. Thanks to its structure and the proof-of-work required to create a new block, a blockchain is inherently resistant to modification of the data. A simple example of blockchain is shown in Figure 2.

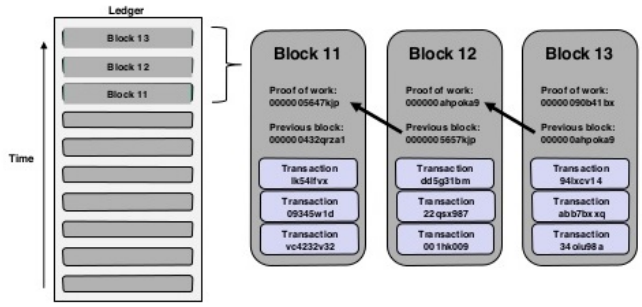

Figure 2. Example of blockchain. Transactions are stored in the blocks of the Ledger. Blocks are linked by hashes.

The presented system relies on Ethereum, an open-source, public, blockchain-based distributed computing platform, which allows to execute smart contracts, immutable programs always visible from the community.

3.1. Ethereum

Ethereum [15] is described as a transactional singleton machine with shared-state. The distributed ledger can be view as a distributed virtual machine which records transactions and state changes. The blockchain is kept alive by the miners, nodes which have three main tasks:

- Verifing the transactions being sent, through the consensum algorithm which is similar to other blockchains, removing the need of central authority;
- Checking if the sender has sufficient gas, the Ethereum's native cryptocurrency, to transfer to the receiver;
- Running the function called by the sender and thus to modify the blockchain state accordingly.

In regards to the third point, we see the most important feature of Ethereum: encoding smart contracts.

3.1.1. Smart Contracts. Smart contracts describe functions, which can be called by nodes participating in the blockchain, and states which can be changed by the nodes. The functions run in the Ethereum Virtual Machine (EVM) which is described as a *quasi*-Turing complete computer able to execute code of any complexity. The *quasi* qualification comes from the fact that the computation is intrinsically bounded to *gas*, which limits the total amount of computation done. If the gas transferred to the miner for the function execution is less than the required amount, then the transaction is not mined.
The contract can use memory and storage to save data. It can use any amount of memory (paying gas) during executing its code, but when execution stops, the entire content of the memory is wiped. The storage on the other hand is persisted into the blockchain itself. It can be changed, but all the changes will be recorded in the ledger.
In the presented sytem, thanks of its persistence and (im)mutability, storage will be used to store and share contents.

3.2. Benefits

In this paragraph, the benefits of using blockchain are discussed. Building the content delivery system on the blockchain it will inherit all the benefits.

- askjda https://www.entrepreneur.com/article/306420 contenuto...

4. System Design

4.1. Overview

4.2. Server Blockchain

4.3. Client Blockchain

4.4. Benefits and Drawbacks

5. Implementation

6. Future Works

7. Conclusion

The conclusion goes here.

Acknowledgments

The authors would like to thank...

References

[1] M. J. Freedman, E. Freudenthal, and D. Mazieres, "Democratizing content publication with coral." in *NSDI*, vol. 4, 2004, pp. 18–18.

[2] L. Wang, V. Pai, and L. Peterson, "The effectiveness of request redirection on cdn robustness," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 345–360, 2002.

[3] R. J. Bayardo and J. Sorensen, "Merkle tree authentication of http responses," in *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 2005, pp. 1182–1183.

[4] A. Adelsbach, M. Rohe, and A.-R. Sadeghi, "Towards multilateral secure digital rights distribution infrastructures," in *Proceedings of the 5th ACM workshop on Digital rights management*. ACM, 2005, pp. 45–54.

[5] C.-H. Chi and Y. Wu, "An xml-based data integrity service model for web intermediaries," in *Proc. 7th IWCW*, 2002.

[6] H. K. Orman, "Data integrity for mildly active content," in *Active Middleware Services, 2001. Third Annual International Workshop on*. IEEE, 2001, pp. 73–77.

[7] P. Maniatis, D. S. Rosenthal, M. Roussopoulos, M. Baker, T. J. Giuli, and Y. Muliadi, "Preserving peer replicas by rate-limited sampled voting," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 44–59.

[8] F. Qin, J. Tucek, J. Sundaresan, and Y. Zhou, "Rx: treating bugs as allergies—a safe method to survive software failures," in *Acm sigops operating systems review*, vol. 39, no. 5. ACM, 2005, pp. 235–248.

[9] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham, "Vigilante: End-to-end containment of internet worms," in *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5. ACM, 2005, pp. 133–147.

[10] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla, "Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems," in *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5. ACM, 2005, pp. 1–16.

[11] N. Michalakis, R. Soulé, and R. Grimm, "Ensuring content integrity for untrusted peer-to-peer content distribution networks," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*. USENIX Association, 2007, pp. 11–11.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[13] R. Sundararajan and S. K. Shukla, "Online identity and authentication using blockchain."

[14] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Open and Big Data (OBD), International Conference on*. IEEE, 2016, pp. 25–30.

[15] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.